

SYSTEM AND METHOD FOR EXECUTING CONDITIONAL BRANCH
INSTRUCTIONS IN A DATA PROCESSOR

CROSS-REFERENCE TO RELATED APPLICATIONS

The present invention is related to those disclosed in the
5 following United States Patent Applications:

- 1) Serial No. [Docket No. 00-BN-051], filed
concurrently herewith, entitled "SYSTEM AND METHOD FOR
EXECUTING VARIABLE LATENCY LOAD OPERATIONS IN A DATA
PROCESSOR";
- 10 2) Serial No. [Docket No. 00-BN-052], filed
concurrently herewith, entitled "PROCESSOR PIPELINE STALL
APPARATUS AND METHOD OF OPERATION";
- 3) Serial No. [Docket No. 00-BN-053], filed
concurrently herewith, entitled "CIRCUIT AND METHOD FOR
15 HARDWARE-ASSISTED SOFTWARE FLUSHING OF DATA AND
INSTRUCTION CACHES";
- 4) Serial No. [Docket No. 00-BN-054], filed
concurrently herewith, entitled "CIRCUIT AND METHOD FOR
SUPPORTING MISALIGNED ACCESSES IN THE PRESENCE OF
20 SPECULATIVE LOAD INSTRUCTIONS";
- 5) Serial No. [Docket No. 00-BN-055], filed

concurrently herewith, entitled "BYPASS CIRCUITRY FOR USE
IN A PIPELINED PROCESSOR";

6) Serial No. [Docket No. 00-BN-057], filed
concurrently herewith, entitled "SYSTEM AND METHOD FOR
ENCODING CONSTANT OPERANDS IN A WIDE ISSUE PROCESSOR";

7) Serial No. [Docket No. 00-BN-058], filed
concurrently herewith, entitled "SYSTEM AND METHOD FOR
SUPPORTING PRECISE EXCEPTIONS IN A DATA PROCESSOR HAVING
A CLUSTERED ARCHITECTURE";

8) Serial No. [Docket No. 00-BN-059], filed
concurrently herewith, entitled "CIRCUIT AND METHOD FOR
INSTRUCTION COMPRESSION AND DISPERSAL IN WIDE-ISSUE
PROCESSORS";

9) Serial No. [Docket No. 00-BN-066], filed
concurrently herewith, entitled "SYSTEM AND METHOD FOR
REDUCING POWER CONSUMPTION IN A DATA PROCESSOR HAVING A
CLUSTERED ARCHITECTURE"; and

10) Serial No. [Docket No. 00-BN-067], filed concurrently herewith, entitled "INSTRUCTION FETCH APPARATUS FOR WIDE ISSUE PROCESSORS AND METHOD OF OPERATION".

5 The above applications are commonly assigned to the assignee of the present invention. The disclosures of these related patent applications are hereby incorporated by reference for all purposes as if fully set forth herein.

TECHNICAL FIELD OF THE INVENTION

The present invention is generally directed to data processors and, more specifically, to a data processor capable of executing conditional branch instructions in a data processor.

BACKGROUND OF THE INVENTION

The demand for high performance computers requires that state-of-the-art microprocessors execute instructions in the minimum amount of time. A number of different approaches have been taken to decrease instruction execution time, thereby increasing processor throughput. One way to increase processor throughput is to use a pipeline architecture in which the processor is divided into separate processing stages that form the pipeline. Instructions are broken down into elemental steps that are executed in different stages in an assembly line fashion.

A pipelined processor is capable of executing several different machine instructions concurrently. This is accomplished by breaking down the processing steps for each instruction into several discrete processing phases, each of which is executed by a separate pipeline stage. Hence, each instruction must pass

sequentially through each pipeline stage in order to complete its execution. In general, a given instruction is processed by only one pipeline stage at a time, with one clock cycle being required for each stage. Since instructions use the pipeline stages in the same order and typically only stay in each stage for a single clock cycle, an N stage pipeline is capable of simultaneously processing N instructions. When filled with instructions, a processor with N pipeline stages completes one instruction each clock cycle.

The execution rate of an N-stage pipeline processor is theoretically N times faster than an equivalent non-pipelined processor. A non-pipelined processor is a processor that completes execution of one instruction before proceeding to the next instruction. Typically, pipeline overheads and other factors decrease somewhat the execution advantage rate that a pipelined processor has over a non-pipelined processor.

An exemplary seven stage processor pipeline may consist of an address generation stage, an instruction fetch stage, a decode stage, a read stage, a pair of execution (E1 and E2) stages, and a write (or write-back) stage. In addition, the processor may have an instruction cache that stores program instructions for execution, a data cache that temporarily stores data operands that otherwise are stored in processor memory, and a register file that

also temporarily stores data operands.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995

The address generation stage generates the address of the next instruction to be fetched from the instruction cache. The instruction fetch stage fetches an instruction for execution from the instruction cache and stores the fetched instruction in an instruction buffer. The decode stage takes the instruction from the instruction buffer and decodes the instruction into a set of signals that can be directly used for executing subsequent pipeline stages. The read stage fetches required operands from the data cache or registers in the register file. The E1 and E2 stages perform the actual program operation (e.g., add, multiply, divide, and the like) on the operands fetched by the read stage and generates the result. The write stage then writes the result generated by the E1 and E2 stages back into the data cache or the register file.

Assuming that each pipeline stage completes its operation in one clock cycle, the exemplary seven stage processor pipeline takes seven clock cycles to process one instruction. As previously described, once the pipeline is full, an instruction can theoretically be completed every clock cycle.

The throughput of a processor also is affected by the size of the instruction set executed by the processor and the resulting

complexity of the instruction decoder. Large instruction sets require large, complex decoders in order to maintain a high processor throughput. However, large complex decoders tend to increase power dissipation, die size and the cost of the processor.

5 The throughput of a processor also may be affected by other factors, such as exception handling, data and instruction cache sizes, multiple parallel instruction pipelines, and the like. All of these factors increase or at least maintain processor throughput by means of complex and/or redundant circuitry that simultaneously increases power dissipation, die size and cost.

10 In many processor applications, the increased cost, increased power dissipation, and increased die size are tolerable, such as in personal computers and network servers that use x86-based processors. These types of processors include, for example, Intel Pentium™ processors and AMD Athlon™ processors. However, in many applications it is essential to minimize the size, cost, and power requirements of a data processor. This has led to the development of processors that are optimized to meet particular size, cost and/or power limits. For example, the recently developed Transmeta Crusoe™ processor reduces the amount of power consumed by the
15
20 processor when executing most x86 based programs. This is particularly useful in laptop computer applications. Other types

of data processors may be optimized for use in consumer appliances (e.g., televisions, video players, radios, digital music players, and the like) and office equipment (e.g., printers, copiers, fax machines, telephone systems, and other peripheral devices).

5 In general, an important design objective for data processors used in consumer appliances and office equipment is the minimization of cost and complexity of the data processor. One way to minimize cost and complexity is to exclude from the processor core functions that can be implemented with memory-mapped peripherals external to the core. For example, cache flushing may be performed using a small memory-mapped device controlled by a specialized software function. The cost and complexity of a data processor may also be minimized by implementing extremely simple exception behavior in the processor core.

10
15
20 As noted above, a wide-issue processor pipeline executes bundles of operations in multiple stages. In a wide-issue processor, multiple concurrent operations are bundled into a single instruction and are issued and executed as a unit. In a clustered architecture, the machine resources are divided into clusters where each cluster consists of one or more register files each of which is associated with a subset of the execution units of the data processor. Communication between clusters is generally restricted,

which presents a significant problem when executing branch instructions - - instructions requiring the "jumps" within program execution steps. In such clusters, branch conditions require large amounts of replicated processing resources or an abundance of global communication wires. Once implemented, such processors are commonly rigid, which precludes any reasonable degree of scalability in the branching architecture.

Two architectures that include partitioned register files, address the foregoing problem in different ways. First, there is the Multiflow Trace architecture which allows multiple branches per cycle (or multi-way branches). This implementation requires that each cluster have its own branch unit that uses local conditions and targets, as well as a global controller, to select a final next program counter address. One major disadvantage of the Multiflow Trace architecture is a requirement of large global communication to perform a branch, which detrimentally impacts both speed and solution cost. Another major disadvantage of the Multiflow Trace architecture is that it is not reasonably possible to use data in one cluster to trigger a branch in another cluster.

Second, there is the Texas Instruments TMS3420C6000 architecture, which allows one branch per cluster (with restrictions). However, multiple branches in one bundle cause

undefined behavior when more than one branch condition is a "true" condition. In other words, the Texas Instruments TMS3420C6000 architecture only supports single-way branches that can be executed on any cluster. This has disadvantages similar to the Multiflow
5 Trace architecture, namely, long connection paths, need to move branch targets to a "global controller," etc.

Therefore, there is a need in the art for improved data processors in which the cost and complexity of the processor core is minimized while maintaining the processor throughput. In
10 particular, there is a need for improved systems and methods for executing conditional branch instructions in a data processor. More particularly, there is a need for systems and methods capable of addressing the problem of using remote branch conditions, while maintaining a local branch address computation, avoiding large
15 amounts of global communication, and enabling a relatively good degree of scalability in the branch architecture.

SUMMARY OF THE INVENTION

To address the above-discussed deficiencies of the prior art, it is a primary object of the present invention to provide a data processor having a clustered architecture and that comprises at least one branching cluster, a plurality of non-branching clusters and remote conditional branching control circuitry. Broadly, the data processor operates to (i) keep program counter ("PC") address computation and, possibly, multiplexing local to the branching cluster, and (ii) compute branch condition (and, possibly, branch priorities in multi-way branching schemes) in any cluster and communicate branch conditions to the branching cluster when the same is computed in a non-branching cluster.

According to an advantageous embodiment, each cluster is capable of computing branch conditions, though only a branching cluster(s) is operable to perform branch address computations. The remote conditional branching control circuitry, which is associated with each of the clusters, is operable in response to sensing a conditional branch instruction in a non-branching cluster to (i) cause the branching cluster to compute a branch address and a next program counter address, (ii) cause the non-branching cluster to compute a branch condition, and (iii) communicate the computed

branch condition from the non-branching cluster to the branching cluster. The data processor then uses the computed branch condition to select one of the branch address and the next program counter address.

5 Preferably, the foregoing may suitably be accomplished, at least in part, through the issuance of a shadow branch instruction in the branching cluster corresponding to the conditional branch instruction existing in the non-branching cluster. An important aspect of this embodiment is that it is possible to optimize for speed while avoiding relatively long and slow global communication delays for PC targets. Another related aspect is that required amounts of communication wires are suitably minimized.

10
15
20

According to one embodiment of the present invention, each of the clusters comprises an instruction execution pipeline comprising N processing stages, each of the N processing stages is capable of performing at least one of a plurality of execution steps associated with a pending instruction being executed by the instruction execution pipeline. According to a related embodiment of the present invention, each of the clusters comprises at least one register file.

According to another embodiment of the present invention, the remote conditional branching control circuitry further causes the

branching cluster to perform a next program counter address computation in response to sensing a conditional branch instruction in the non-branching cluster. According to a related embodiment of the present invention, the remote conditional branching control circuitry selects one of the computed next program counter address and the computed branch address in response to the value of the computed branch condition. In a further related embodiment of the present invention, the remote conditional branching control circuitry comprises a multiplexor that is responsive to the computed branch condition.

The foregoing has outlined rather broadly the features and technical advantages of the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features and advantages of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art should appreciate that they may readily use the conception and the specific embodiment disclosed as a basis for modifying or designing other structures for carrying out the same purposes of the present invention. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the invention in its broadest form.

Before undertaking the DETAILED DESCRIPTION OF THE INVENTION below, it may be advantageous to set forth definitions of certain words and phrases used throughout this patent document: the terms "include" and "comprise," as well as derivatives thereof, mean inclusion without limitation; the term "or," is inclusive, meaning and/or; the phrases "associated with" and "associated therewith," as well as derivatives thereof, may mean to include, be included within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, juxtapose, be proximate to, be bound to or with, have, have a property of, or the like; and the term "controller" and "circuitry" means any device, system or part thereof that controls at least one operation, such a device, system or part thereof may be implemented in hardware, firmware or software, or some combination of at least two of the same. It should be noted that the functionality associated with any particular controller or circuitry may be centralized or distributed, whether locally or remotely. Definitions for certain words and phrases are provided throughout this patent document, those of ordinary skill in the art should understand that in many, if not most instances, such definitions apply to prior, as well as future uses of such defined words and phrases.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like numbers designate like objects, and in which:

FIGURE 1 illustrates a block diagram of a processing system that contains a data processor in accordance with the principles of the present invention;

FIGURE 2 illustrates the exemplary data processor in greater detail according to one embodiment of the present invention;

FIGURE 3 illustrates a cluster in the exemplary data processor according to one embodiment of the present invention;

FIGURE 4 illustrates the operational stages of the exemplary data processor according to one embodiment of the present invention;

FIGURE 5 illustrates an exemplary data processor having a branching cluster and three non-branching clusters according to one embodiment of the present invention;

FIGURE 6 illustrates a block diagram of exemplary next program computation circuitry according to one embodiment of the present invention;

FIGURE 7 illustrates a conceptual diagram of remote conditional branching control circuitry according to one embodiment of the present invention; and

FIGURE 8 illustrates a flow diagram of an exemplary method of operating a data processor according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

FIGURES 1 through 8, discussed below, and the various embodiments used to describe the principles of the present invention in this patent document are by way of illustration only and should not be construed in any way to limit the scope of the invention. Those skilled in the art will understand that the principles of the present invention may be implemented in any suitably arranged data processor supporting a clustered architecture.

FIGURE 1 is a block diagram of processing system 10, which contains data processor 100 in accordance with the principles of the present invention. Data processor 100 comprises processor core 105 and N memory-mapped peripherals interconnected by system bus 120. The N memory-mapped peripherals include exemplary memory-mapped peripherals 111-114, which are arbitrarily labeled Memory-Mapped Peripheral 1, Memory-Mapped Peripheral 2, Memory-Mapped Peripheral 3, and Memory-Mapped Peripheral N. Processing system 10 also comprises main memory 130. In an advantageous embodiment of the present invention, main memory 130 may be subdivided into program memory 140 and data memory 150.

The cost and complexity of data processor 100 is minimized by

excluding from processor core 105 complex functions that may be implemented by one or more of memory-mapped peripherals 111-114. For example, memory-mapped peripheral 111 may be a video codec and memory-mapped peripheral 112 may be an audio codec. Similarly, memory-mapped peripheral 113 may be used to control cache flushing. The cost and complexity of data processor 100 is further minimized by implementing extremely simple exception behavior in processor core 105, as explained below in greater detail.

Processing system 10 is shown in a general level of detail because it is intended to represent any one of a wide variety of electronic devices, particularly consumer appliances. For example, processing system 10 may be a printer rendering system for use in a conventional laser printer. Processing system 10 also may represent selected portions of the video and audio compression-decompression circuitry of a video playback system, such as a video cassette recorder or a digital versatile disk (DVD) player. In another alternative embodiment, processing system 10 may comprise selected portions of a cable television set-top box or a stereo receiver. The memory-mapped peripherals and a simplified processor core reduce the cost of data processor 100 so that it may be used in such price sensitive consumer appliances.

In the illustrated embodiment, memory-mapped peripherals 111-

114 are shown disposed within data processor 100 and program memory 140 and data memory 150 are shown external to data processor 100. It will be appreciated by those skilled in the art that this particular configuration is shown by way of illustration only and should not be construed so as to limit the scope of the present invention in any way. In alternative embodiments of the present invention, one or more of memory-mapped peripherals 111-114 may be externally coupled to data processor 100. Similarly, in another embodiment of the present invention, one or both of program memory 140 and data memory 150 may be disposed on-chip in data processor 100.

FIGURE 2 is a more detailed block diagram of exemplary data processor 100 according to one embodiment of the present invention. Data processor 100 comprises instruction fetch cache and expansion unit (IFCEXU) 210, which contains instruction cache 215, and a plurality of clusters, including exemplary clusters 220-222. Exemplary clusters 220-222 are labeled Cluster 0, Cluster 1 and Cluster 2, respectively. Data processor 100 also comprises core memory controller 230 and interrupt and exception controller 240.

A fundamental object of the design of data processor 100 is to exclude from the core of data processor 100 most of the functions that can be implemented using memory-mapped peripherals external to

the core of data processor 100. By way of example, in an exemplary embodiment of the present invention, cache flushing may be efficiently accomplished using software in conjunction with a small memory-mapped device. Another object of the design of data processor 100 is to implement a statically scheduled instruction pipeline with an extremely simple exception behavior.

Clusters 220-222 are basic execution units that comprise one or more arithmetic units, a register file, an interface to core memory controller 230, including a data cache, and an inter-cluster communication interface. As will be described in greater detail hereafter, it is preferable that at least one of clusters 220-222 is a branching cluster (for instance cluster 220) while the remaining clusters are non-branching clusters. Each cluster 220-222 is capable of computing branch conditions, though only branching cluster 220 is operable to perform branch address computations. According to the present embodiment, data processor 100 operates, in response to sensing a conditional branch instruction in a non-branching cluster, to (i) cause branching cluster 220 to compute a branch address and a next program counter address, (ii) cause non-branching clusters 221-222 to compute a branch condition, and (iii) communicate the computed branch condition from non-branching clusters 221-222 to branching cluster

220. Data processor 100 then uses the computed branch condition to select one of the branch address and the next program counter address.

Because conventional processor cores can execute multiple
5 simultaneously issued operations, the traditional word
"instruction" is hereby defined with greater specificity. For the
purposes of this disclosure, the following terminology is adopted.
An "instruction" or "instruction bundle" is a group of
simultaneously issued operations encoded as "instruction
10 syllables". Each instruction syllable is encoded as a single
machine word. Each of the operations constituting an instruction
bundle may be encoded as one or more instruction syllables.
Hereafter, the present disclosure may use the shortened forms
"instruction" and "bundle" interchangeably and may use the
15 shortened form "syllable." In an exemplary embodiment of the
present invention, each instruction bundle consists of 1 to 4
instruction syllables. Flow control operations, such as branch or
call, are encoded in single instruction syllables.

FIGURE 3 is a more detailed block diagram of branching
20 cluster 220 in data processor 100 according to one embodiment of
the present invention. Branching cluster 220 comprises instruction
buffer 305, register file & rewindable register buffer unit 310,

program counter and branch unit 315, instruction decoder 320, load store unit 325, data cache 330, integer units 341-344, and multipliers 351-352. Cluster 220 is implemented as an instruction pipeline.

5 Instructions are issued to an operand read stage associated with register file 310 and then propagated to the execution units (i.e., integer units 341-244, multipliers 351-352). Exemplary cluster 220 accepts one bundle comprising one to four syllables in each cycle. The bundle may consist of any combination of four integer operations, two multiplication operations, and one memory operation (i.e., read or write) and one branch operation. Operations that require long immediates (constants) require two syllables.

10 In specifying a cluster, it is assumed that no instruction bits are used to associate operations with functional units. For example, arithmetic or load/store operations may be placed in any of the four words encoding the operations for a single cycle. This may require imposing some addressing alignment restrictions on multiply operations and long immediates (constants).

15 This following describes the architectural (programmer visible) status of the core of data processor 100. One design objective of data processor 100 is to minimize the architectural

status. All non-user visible status information resides in a memory map, in order to reduce the number of special instructions required to access such information. While each of the clusters 220-222 is capable of computing branch conditions, only branching
5 cluster 220 is operable to perform branch address computations.

Program Counter

In an exemplary embodiment of the present invention, the program counter (PC) in program counter and branch unit 315 is a 32-bit byte address pointing to the beginning of the current instruction bundle in memory. The two least significant bits (LSBs) of the program counter are always zero. In operations that assign a value to the program counter, the two LSBs of the assigned value are ignored.

According to the illustrated embodiment, when a conditional branch instruction is executing in one of non-branching clusters 221-222, the program counter is operable to compute a branch address and a next program counter address. The non-branching cluster 221-222 executing the conditional branch instruction computes a branch condition and communicates the same to the
15 program counter. The program counter includes an input channel associated therewith to receive the computed branch condition, and, in response thereto, to select one of the branch address or the
20

next program counter address.

Register File 310

In an exemplary embodiment, register file 310 contains 64 words of 32 bits each. Reading Register 0 (i.e., R0) always returns the value zero.

Link Register

Register 63 (i.e., R63) is used to address the link register by the call and return instructions. The link register (LR) is a slaved copy of the architecturally most recent update to R63. R63 can be used as a normal register, between call and return instructions. The link register is updated only by writes to R63 and the call instruction. At times the fact that the link register is a copy of R63 and not R63 itself may be visible to the programmer. This is because the link register and R63 get updated at different times in the pipeline. Typically, this occurs in the following cases:

1) ICALL and IGOTO instructions - Since these instructions are executed in the decode stage, these operations require that R63 be stable. Thus, R63 must not be modified in the instruction bundle preceding one of these operations. Otherwise unpredictable results may occur in the event of an interrupt; and

2) An interrupt or exception may update the link register incorrectly. Thus, all interrupt and exception handlers must explicitly write R63 prior to using the link register through the execution of an RFI, ICALL or IGOTO instruction. This requirement
5 can be met with a simple MOV instruction from R63 to R63.

Branch Bit File

The branch architecture of data processor 100 uses a set of eight (8) branch bit registers (i.e., B0 through B7) that may be read or written independently. In an exemplary embodiment of the present invention, data processor 100 requires at least one instruction to be executed between writing a branch bit and using the result in a conditional branch operation.

Control Registers

A small number of memory mapped control registers are part of the architectural state of data processor 100. These registers include support for interrupts and exceptions, and memory protection.

The core of data processor 100 is implemented as a pipeline that requires minimal instruction decoding in the early pipeline stages. One design objective of the pipeline of data processor 100 is that it support precise interrupts and exceptions. Data processor 100 meets this objective by updating architecturally
20

visible state information only during a single write stage. To accomplish this, data processor 100 makes extensive use of register bypassing circuitry to minimize the performance impact of meeting this requirement.

FIGURE 4 is a block diagram illustrating the operational stages of pipeline 400 in exemplary data processor 100 according to one embodiment of the present invention. In the illustrated embodiment, the operational stages of data processor 100 are address generation stage 400, fetch stage 402, decode stage 403, read stage 404, first execution (E1) stage 405, second execution (E2) stage 406 and write stage 407.

Address Generation Stage 401 and Fetch Stage 402

Address generation stage 401 comprises a fetch address generator 410 that generates the address of the next instruction to be fetched from instruction cache 215. Fetch address generator 410 receives inputs from exception generator 430 and program counter and branch unit 315. Fetch address generator 410 generates an instruction fetch address (FADDR) that is applied to instruction cache 215 in fetch stage 402 and to an instruction protection unit (not shown) that generates an exception if a protection violation is found. Any exception generated in fetch stage 402 is postponed to write stage 407. Instruction buffer 305 in fetch stage 402

receives instructions as 128-bit wide words from instruction cache 215 and the instructions are dispatched to the cluster.

Decode Stage 403

Decode stage 403 comprises instruction decode block 415 and program counter (PC) and branch unit 315. Instruction decode block 415 receives instructions from instruction buffer 305 and decodes the instructions into a group of control signals that are applied to a execution units in E1 stage 405 and E2 stage 406. According to the illustrated embodiment, when a conditional branch instruction is sensed in the execution pipeline of a non-branching cluster 221-222, data processor 100 also issues a shadow conditional branch instruction in branching cluster 220 causing program counter and branch unit 315 to perform a branch address computation as well as a next program counter address computation, thereby enabling program counter and branch unit 315 to evaluate branch instructions detected within the 128-bit wide words.

The non-branching cluster executing the conditional branching instruction computes a branch condition and circuitry associating clusters 220-221 communicates the computed branch condition from the non-branching cluster to branching cluster 220. An important aspect of this implementation is that a taken branch incurs a one cycle delay and the instruction being incorrectly fetched while the

branch instruction is evaluated is discarded.

Read Stage 404

In read stage 404, operands are generated by register file access, bypass and immediate (constant) generation block 420. The sources for operands are the register files, the constants (immediates) assembled from the instruction bundle, and any results bypassed from operations in later stages in the instruction pipeline.

E1 Stage 405 and E2 Stage 406

The instruction execution phase of data processor 100 is implemented as two stages, E1 stage 405 and E2 stage 406 to allow two cycle cache access operations and two cycle multiplication operations. Exemplary multiplier 351 is illustrated straddling the boundary between E1 stage 405 and E2 stage 406 to indicate a two cycle multiplication operation. Similarly, load store unit 325 and data cache 330 are illustrated straddling the boundary between E1 stage 405 and E2 stage 406 to indicate a two cycle cache access operation. Integer operations are performed by integer units, such as IU 341 in E1 stage 405. Exceptions are generated by exception generator 430 in E2 stage 406.

Results from fast operations are made available after E1 stage 405 through register bypassing operations. An important

architectural requirement of data processor 100 is that if the results of an operation may be ready after E1 stage 405, then the results are always ready after E1 stage 405. In this manner, the visible latency of operations in data processor 100 is fixed.

5 Write Stage 407

At the start of write stage 407, any pending exceptions are raised and, if no exceptions are raised, results are written by register write back and bypass block 440 into the appropriate register file and/or data cache location. In data processor 100, write stage 407 is the "commit point" and operations reaching write stage 407 in the instruction pipeline and not "excepted" are considered completed. Previous stages (i.e., address generation, fetch, decode, read, E1, E2) are temporally prior to the commit point. Therefore, operations in address generation stage 401, fetch stage 402, decode stage 403, read stage 404, E1 stage 405 and E2 stage 406 are flushed when an exception occurs and are acted upon in write stage 407.

Load operations that transfer data from data cache 330 to the register files are performed in E1 stage 405, E2 stage 406, and write stage 407. Data shifting is performed early in write stage 407 prior to loading the data into the appropriate register file in register write back and bypass block 440. In order to

maximize processor throughput, the present invention implements bypassing circuitry in the pipeline that permits data from load word operations to bypass the shifting circuitry in write stage 407.

5 FIGURE 5 illustrates a data processor 100 having a clustered architecture according to one embodiment of the present invention. For purposes of illustration, concurrent reference is implicitly made to the exemplary embodiments of FIGURES 1 to 4.

10 Exemplary data processor 100 illustratively includes a branching cluster 220 and three non-branching clusters 221-223, each of which is capable of computing branch conditions. Each cluster 220-223 comprises an instruction execution pipeline comprising N processing stages, wherein each processing stage is capable of performing at least one of a plurality of execution steps associated with a pending instruction being executed by the instruction execution pipeline.

15 According to the present embodiment, exemplary branching cluster 220 includes program counter and branch unit 315 that illustratively includes next PC computation circuitry 500.
20 Exemplary next PC computation circuitry 500 is operable to determine the address of the next instruction to be executed by data processor 100. When an instruction executing in a non-

branching cluster 221 is a conditional branching instruction, branching cluster 220, via program counter and branch unit 315 and next PC computation circuitry 500, computes both a next program counter address and a branch address. The foregoing is accomplished while non-branching cluster 221 computes the condition and communicates the computed branch condition from non-branching cluster 221 to branching cluster 220. Next PC computation circuitry 500 then selects among one of the computed next program counter address and the computed branch address in response to the received computed branch condition.

According to the present embodiment, non-branching cluster 221, branching cluster 220 (particularly, program counter and branch unit 315 and next PC computation circuitry 500), and the wires associating the same cooperate to form remote conditional branching control circuitry that causes branching cluster 220 to perform a branch address computation in response to sensing a conditional branch instruction in non-branching cluster 221, and to communicate a computed branch condition from non-branching cluster 221 to branching cluster 220. This is accomplished, according to this embodiment, by issuing a shadow conditional branch instruction in branching cluster 220 to perform branch address computation in response to sensing the conditional branch instruction in non-

branching cluster 221. In a multi-cluster environment it is advantageous to begin address computation as early in the instruction execution pipeline as possible. The implementation described here performs the next PC computation on one branching cluster, but uses condition information from at least one of the non-branching clusters.

In particular, a conditional branch instruction that requires condition data from a cluster other than branching cluster 220 causes issuance of two identical branches - one on cluster 220 and one on the cluster providing the condition data, cluster 221. A priority encoder may suitably be used to give precedence to conditions other than cluster 220. Thus, the only information that needs communication between clusters 220 and 221 and program counter and branch unit 315 is condition data. This may suitably be encoded, for example, with two signals per cluster - - data and valid.

An important aspect of this invention is that the principles thereof may suitably be extended to support multiple branches per cycle (multi-way branches) by exploiting multiple-issue capability of branching cluster 220. By way of example, if branching cluster 220 allows four instructions per cycle, branching cluster 220 may suitably support a four-way branch per cycle with the same

technique. As such, other clusters (non-branching clusters, branching clusters, or both) can participate (i.e., non-branching cluster/other branching cluster cooperates with branching cluster 220) to the branch conditions (and priorities) by sending the appropriate condition bits to branching cluster 220.

In this way, FIGURE 5 may also illustrate an exemplary multi-way branching structure wherein a clustered architecture is shown with four clusters that can execute a two-way branch per cycle. Consider the following code:

```
c1 cmp $b1.1 = ...
c3 cmp $b3.1 = ...
;;
;;
c0, c1 br L0, $b1, 1
c0, c3 br L1, $b3, 1
;;
```

Table 1

This code example shows execution of a 2-way branch using compare conditions from Clusters 221 and 223 and two instruction slots in branching cluster 220 for branch targets. The notation "c0,c1 br L0, \$b1,1" indicates that two syllables are used: one syllable in cluster 220 to produce the branch target (and start the address computation) and one syllable in cluster 221 to send the condition register (in this case \$b1.1) to branching cluster 220. Some delay (in the example: 1 extra cycle) may be necessary between the producer of the compare conditions and the branches. However this delay is exposed at the architecture level, and the compiler can apply known scheduling techniques to hide it when possible.

FIGURE 6 illustrates a block diagram of exemplary next PC computation circuitry 500 according to one embodiment of the present invention. Next PC computation circuitry 500 illustratively includes an adder circuit 600, a counter circuit 605 and a multiplexor 610. Each of adder circuit 600 and counter circuit 605 receive the current program counter as an input. Adder circuit 600

also receives an offset value (for branching) as another input. Multiplexor 610 receives as inputs the outputs of each of adder circuit 600 and counter circuit 605, and operates to select one of the same as a function of a condition signal.

5 By way of discussion, next PC computation circuitry 500 computes next program counter addresses. Commonly this is accomplished using counter circuit 605 to simply determine the next program counter address. However, when a conditional branch instruction is executing in one of non-branching clusters 221-223, program counter and branch unit 315 is operable to compute both a branch address and a next program counter address, as above-described. It is clear that the branch address calculation, which requires performing an addition, can proceed in parallel with the condition computation in the non-branching cluster. Furthermore, the application of the condition to the calculation consists purely of setting up a multiplexor - inherently a fast operation.

FIGURE 7 illustrates a conceptual diagram of remote conditional branching control circuitry (generally designated 700) according to one embodiment of the present invention. Exemplary remote conditional branching control circuitry 700 illustratively includes a branching cluster 220 and a non-branching cluster 221. Exemplary branching cluster 220 illustratively includes program

counter and branch unit 315 that illustratively includes next PC computation circuitry 500. Exemplary non-branching cluster 221 illustratively includes fetch stage 402, decode stage 403, register file 420 and execution stage 405; 406.

5 For purposes of illustration, the functionality of remote conditional branching control circuitry 700 is described with concurrent reference to FIGURE 8. FIGURE 8 illustrates a flow diagram (generally designated 800) of an exemplary method of operating data processor 100 when a conditional branch instruction is executing in a non-branching cluster according to one embodiment of the present invention.

To begin, data processor 100 enters fetch stage 402 first, generating an instruction fetch address (process step 805; FADDR) and then enters decode stage 403 second. During decode stage 403, instruction buffer 305 of FIGURE 3 receives instructions as 128-bit wide words from instruction cache 215 and the instructions are dispatched to a cluster 220-222 of FIGURE 2 (process step 810).

10
15
20 According to the illustrated embodiment, when a conditional branch instruction is sensed in the execution pipeline of a non-branching cluster 221 ("Y" branch of decision step 815), data processor 100 issues a shadow conditional branch instruction in branching cluster 220 causing program counter and branch unit 315

to perform a branch address computation as well as a next program counter address computation (process step 820), thereby enabling program counter and branch unit 315 via next PC computation circuitry 500 to evaluate branch instructions detected within the 128-bit wide words.

Data processor 100 continues through the read stage where operands are generated by register file access, bypass and immediate (constant) generation block 420 (process step 825). The sources for operands are the register files, the constants (immediates) assembled from the instruction bundle, and any results bypassed from operations in later stages in the instruction pipeline.

Data processor 100 continues through the instruction execution stage 405; 406, and non-branching cluster 221 executing the conditional branching instruction computes a branch condition and circuitry associating clusters 220-221 communicates the computed branch condition from non-branching cluster 221 to branching cluster 220 (Process step 830). According to this implementation, a taken branch incurs a one cycle delay and the instruction being incorrectly fetched while the branch instruction is evaluated is discarded.

Importantly, conditional branch instructions require the

calculation of a condition to decide whether the branch should be taken and a destination address. In a clustered architecture, it is advantageous if the branch condition can be computed on any cluster 220-223 to eliminate movement of data between clusters. In contrast, the computation of the destination address frequently consists of adding a constant to current program counter and does not require the ability to perform the calculation on an arbitrary cluster. Traditionally, a difficulty associated with implementing conditional branch instructions is their existence on critical instruction execution paths. This difficulty may suitably be reduced by performing the address calculation in parallel with the computation of the branch condition as above-described.

Lastly, branching cluster 220, via next PC computation circuitry 500 uses the computed branch condition to select one of the branch address or the next program counter address (process step 835). From the foregoing, it is clear that the exemplary branch architecture for a clustered machine illustrates several aspects of the principles hereof, namely, (i) early computation of the branch address minimizes taken branch penalties: if the branch address is computed late in the pipeline, cycles are lost before the instruction at the new address can be fetched - indicating that next PC address computation should be centralized in a single

location; and (ii) issuance of compare operations on multiple clusters because working sets are distributed across the clusters, movement of all branch information to one specific cluster would unnecessarily increase inter-cluster traffic - - branching
5 condition computation should be decentralized to the individual clusters.

Although the present invention has been described in detail, those skilled in the art should understand that they can make various changes, substitutions and alterations herein without departing from the spirit and scope of the invention in its broadest form.

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209